Claims

What is claimed is:

1. A method of defining a trace point, comprising:

   defining a trace point representation in a program source code;

   compiling the program source code to generate an instrumented program comprising the trace point corresponding to the trace point representation; and

   associating the trace point with a placeholder function configured to produce a minimal disabled probe effect.

2. The method of claim 1, further comprising:

   storing an address location of the trace point and a probe handler associated with the address location in a trace point table, wherein the address location and the probe handler are identified by a trace point identifier.

3. The method of claim 2, further comprising:

   obtaining the address location of the trace point; and

   identifying the probe handler associated with the address location.

4. The method of claim 1, wherein the trace point representation comprises a tracing function defined by a code.

5. The method of claim 1, wherein the placeholder function comprises at least one instruction designed to use minimal system resources.

6. The method of claim 5, wherein at least one instruction comprises a no-operation instruction.

7. The method of claim 5, wherein the at least one instruction comprises a first instruction comprising a trap instruction and a second instruction comprising a no-operation instruction.

8. The method of claim 2, further comprising:

obtaining a tracing function name from trace object code using a tracing framework;

determining an address location of the trace point in the instrumented program by accessing the trace function name in the trace point table; and

replacing a placeholder function located at the address location of trace point with a function call into the tracing framework.

9. A method for enabling a trace point, comprising:

obtaining a tracing function name from trace object code using a tracing framework, wherein the tracing function name comprises a probe handler;

determining an address location of the trace point in an instrumented program by accessing the probe handler in a trace point table; and

replacing a placeholder function located at the address location of the trace point with a function call into the tracing framework.

10. The method of claim 9, further comprising:

disabling the trace point by replacing the function call with the placeholder function.

11. The method of claim 9, further comprising:

storing the probe handler and the address location in a trace point table, wherein the trace function name and address location are identified by a trace point identifier.

12. The method of claim 9, further comprising:

accessing a probe provider using the tracing framework; and

directing the probe provider to enable a probe associated with the trace point.

13. The method of claim 9, further comprising:

generating the trace point table during compilation of a program source code.

14. The method of claim 9, wherein the function call comprises calling to a jump instruction to the tracing framework.

15. The method of claim 9, wherein the function call comprises:

generating a trap transferring control to a trap handler associated with the trap;

calling the tracing framework from the trap handler; and

emulating a patched-over instruction.

16. The method of claim 15, wherein the emulating comprises incrementing a saved instruction pointer by one prior to issuing a return instruction from the trap.

17. The method of claim 9, wherein the placeholder function comprises at least one instruction designed to use minimal system resources.

18. The method of claim 17, wherein the at least one instruction comprises a no-operation instruction.

19. The method of claim 17, wherein the at least one instruction comprising a first instruction comprises a trap instruction and a second instruction comprising a no-operation instruction.

20. A computer system on a network for defining a trace point comprising:

a processor;

a memory;

a storage device; and

software instructions stored in the memory for enabling the computer system to:

define a trace point representation in a program source code;

compile the program source code to generate an instrumented program comprising the trace point corresponding to the trace point representation; and

wherein the trace point is associated with a placeholder function configured to produce a minimal disabled probe effect.

21. The computer system of claim 20, further comprising software instructions stored in the memory for enabling the computer system to:

store an address location of the trace point and a probe handler associated with the address location in a trace point table, wherein the address location and the probe handler are identified by a trace point identifier.

22. The computer system of claim 21, further comprising software instructions stored in the memory for enabling the computer system to:

obtain a tracing function name from trace object code using a tracing framework;

determine an address location of the trace point in the instrumented program by accessing the trace function name in the trace point table; and

replace a placeholder function located at the address location of trace point with a function call into the tracing framework.

23. The computer system of claim 20, further comprising software instructions stored in the memory for enabling the computer system to:

obtain the address location of the trace point; and

identify the probe handler associated with the address location.

24. The computer system of claim 20, wherein the trace point representation comprises a tracing function defined by a code.

25. The computer system of claim 20, wherein the placeholder function comprises at least one instruction designed to use minimal system resources.

26. The computer system of claim 25, wherein the at least one instruction comprises a no-operation instruction.

27. The computer system of claim 25, wherein the at least one instruction comprises a first instruction comprising a trap instruction and a second instruction comprising a no-operation instruction.

28. A computer system on a network for enabling a trace point comprising:

a processor;

a memory;

a storage device; and

software instructions stored in the memory for enabling the computer system to:

obtain a tracing function name from trace object code using a tracing framework, wherein the tracing function name comprises a probe handler;

determine an address location of the trace point in an instrumented program by accessing the probe handler in a trace point table; and

replace a placeholder function located at the address location of the trace point with a function call into the tracing framework.

29. The computer system of claim 28, further comprising software instructions stored in the memory for enabling the computer system to:

disable the trace point by replacing the function call with the placeholder function.

30. The computer system of claim 28, further comprising software instructions stored in the memory for enabling the computer system to:

store the probe handler and the address location in a trace point table, wherein the trace function name and address location are identified by a trace point identifier.

31. The computer system of claim 28, further comprising software instructions stored in the memory for enabling the computer system to:

access a probe provider using the tracing framework; and

direct the probe provider to enable a probe associated with the trace point.

32. The computer system of claim 28, further comprising software instructions stored in the memory for enabling the computer system to:

generate the trace point table during compilation of a program source code.

33. The computer system of claim 28, wherein the function call comprises calling to a jump instruction to the tracing framework.

34. The computer system of claim 28, wherein the function call comprises:

generating a trap transferring control to a trap handler associated with the trap;

calling the tracing framework from the trap handler; and

emulating a patched-over instruction.

35. The computer system of claim 34, wherein the emulating comprises incrementing a saved instruction pointer by one prior to issuing a return instruction from the trap.

36. The computer system of claim 28, wherein the placeholder function comprises at least one instruction designed to use minimal system resources.

37. The computer system of claim 36, wherein the at least one instruction comprises a no-operation instruction.

38. The computer system of claim 36, wherein the at least one instruction comprising a first instruction comprises a trap instruction and a second instruction comprising a no-operation instruction.